Teaching Image-Processing Programming for Mobile Devices: A Software Development Perspective

Matthias Trapp[1], Sebastian Pasewaldt[2], Tobias Dürschmid[1], Amir Semmo[1] and Jürgen Döllner[1]

[1]Hasso Plattner Institute, Digital Engineering Faculty, University of Potsdam, Germany
[2]Digital Masterpieces GmbH, Germany

Abstract

In this paper we present a concept of a research course that teaches students in image processing as a building block of mobile applications. Our goal with this course is to teach theoretical foundations, practical skills in software development as well as scientific working principles to qualify graduates to start as fully-valued software developers or researchers. The course includes teaching and learning focused on the nature of small team research and development as encountered in the creative industries dealing with computer graphics, computer animation and game development. We discuss our curriculum design and issues in conducting undergraduate and graduate research that we have identified through four iterations of the course. Joint scientific demonstrations and publications of the students and their supervisors as well as quantitative and qualitative evaluation by students underline the success of the proposed concept. In particular, we observed that developing using a common software framework helps the students to jump start their course projects, while industry software processes such as branching coupled with a three-tier breakdown of project features helps them to structure and assess their progress.

Categories and Subject Descriptors (according to ACM CCS): K.3.2 [Computer Graphics]: Computers and Education—Computer and Information Science Education

1. Introduction

Mobile devices are causing a fundamental shift in how digital content is created, processed and consumed. In particular, with the continuous advancements of mobile camera hardware and processing power, graphics applications—once designed exclusively for desktop systems—have emerged into the ubiquitous domain, for instance non-photorealistic rendering [Dev13]. However, developing mobile applications requires special knowledge that is typically not taught in core courses of computer science, such as dealing with the specific requirements and inherent constraints of memory resources, APIs, and user interfaces [CPAM08]. In this context, this paper presents the experience of teaching real-time image processing techniques and algorithms for mobile devices [TMB14] in *Computer Graphics* subjects within the *Computer Science* degree with a strong focus on the software development process.

Especially in higher education it becomes more and more important to bridge the gaps between theory and practice—between concept and implementation—by increasing software development experiences and thus supporting professionalism and operational readiness of graduates in the software industry. Further, undergraduate research is also considered important by employers. To facilitate both, universities are required to educate graduate software developers with not only vocational but also advanced research skills [AAF16]. This can be achieved by taking the demands of companies into account and adapt the teaching process, which is appreciated by our students, since they articulated an interest in industry projects or collaborations during their education. With respect to this, the presented approach implements such a collaboration between academic institution and companies by the example of image processing algorithms and techniques for both, bachelor and master student courses.

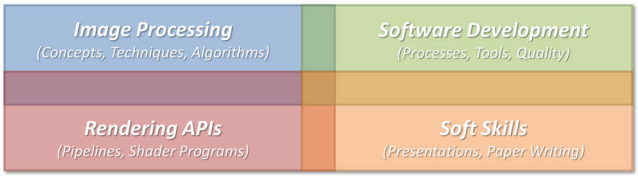

Figure 1: *Overlapping fields of education targeted by the presented project seminar structure.*

Teaching students in image processing techniques for mobile devices comprises a number of different fields (Figure 1). Besides the fundamental knowledge of concepts and data structures specific to image processing, aspects of real-time graphics APIs [FWW13]

become more important when focusing on mobile devices as implementation platform. To successfully handle a topic within a semester term, a student and adviser is confronted with the following aspects and contents of teaching:

Image Processing: This teaches fundamental concepts, techniques, and algorithms of digital image processing [GW06]. The students should understand these independently of possible implementations to apply these in different contexts.

Rendering APIs: This is about basic concepts of OpenGL ES and the OpenGL shading language , standard APIs in industry and education, especially for mobile devices. The goals are to teach fundamental API principles and performance optimization techniques for digital image processing enabling students to transfer their knowledge to hardware-accelerated implementations of image processing pipelines.

Software Development: This teaches foundations of agile software development processes for individual workers and teams and introduces tools for code versioning and build processes. With respect to mobile platforms, we focus on Android application development [Wer13] using Java in combination with the Android NDK for performance critical tasks.

Research Skills: In addition to technical skills, the preparation and performance of presentations, literature work, and writing scientific reports or papers represent major necessities of higher education. Therefore, these take a high priority in the course design.

Covering all these aspect within a project seminar proves unrealistic when starting from scratch. Instead, we rely on a common software framework provided by a collaborating company and integrate it into teaching to facilitate the work on complex topics, by enabling fast prototyping and the reuse of existing software components. However, the educational objective differs from software engineering courses that are oriented on the development process, methodologies, and organization by experiencing a team-based project [DFH02]. In contrast, the presented course focuses on the development of mobile image processing applications in an artifact-oriented context with a single or two programmers. Therefore, rather than teaching software engineering through a specific application domain [CC05], the main goal is to teach image processing programming under conditions close to industrial practice.

To summarize, this paper makes the following contributions: (1) it presents a concept for a series of project seminars with a strong focus on software development that can be implemented both in bachelor and master courses; (2) it gives sufficient implementation details for reproduction, and (3) it presents an evaluation and discussion mainly based on the student's feedback. The remainder of this paper is structured as follows. Section 2 presents the concept and details of our teaching approach. Section 3 discusses the presented concept by means of student examples and course evaluation results. Finally, Section 4 concludes this paper and presents future work.

2. Teaching Concept

2.1. Academic Environment

The presented concept was implemented for two bachelor and two master courses in the Computer Graphics Systems Group (8 to 14 researchers) of the Hasso Plattner Institute at University of Potsdam during the last two years. It is designed for 6 European Credit Points System (ECTS) points per semester term, which relates to 180 h of study time, i.e., approx. 10 h to 12 h per week. In general, it is limited to approximately 8 to 16 participants—without group work—and is supervised by three to four course instructors. The students are required (1) to have successfully participated in basic computer graphics lectures and (2) have basic knowledge in programming (C++, Java). We use the Moodle platform [DT03] as learning management system (LMS). All students are required to use a common software framework [SDT*16] that has been implemented for the Android operating system. It is written in Java and uses OpenGL ES for GPU-accelerated image processing. Furthermore, it uses a software product line architecture [DTD17] to enable students to create their own apps based on a common core. In addition to example apps and reusable software components, the framework includes a collection of effects to facilitate rapid effect creation and ease a low-level to high-level development [She11], e.g., by writing GLSL shader code or combining existing effects. The usage of the framework lets the students focus on the development of computer graphics techniques. Moreover, it trains their skills to integrate solutions into an existing software architecture to gain practical experience.

2.2. Overview of Course Design

Figure 2 illustrates an overview of the general course structure that can be implemented at bachelor and master levels. It basically comprises three phases. Its individual steps are described in the remainder of this section:

Introduction Phase: The course introduction phase usually consists of five to six lecture sessions and is structured into (1) *seminar introduction* (Section 2.3), (2) *topic assignment* (Section 2.4), and (3) *intermediate presentation* (Section 2.5).

Project Phase: Subsequent to the seminar introduction, students start to work on their individual topics supported by regular one-on-one meetings with their respective instructors (Figure 2.6).

Evaluation & Grading Phase: The final phase comprises the student evaluation and grading based on *final presentations* (Section 2.7), submitted *code artifacts* (Section 2.8), and *project documentation* (Section 2.9) in form of a research paper (master courses only).

2.3. Seminar Introduction Sessions

The seminar introduction sessions basically covers the following aspects in the form of separate 90 minutes lectures:

Seminar Kick-Off: This session briefly: (1) introduces the problems and challenges of image processing in general, (2) presents the course topics to choose from, (3) describes the course format and structure, and (4) communicates the assessment scale. In addition thereto, the students proficiency level is recorded by

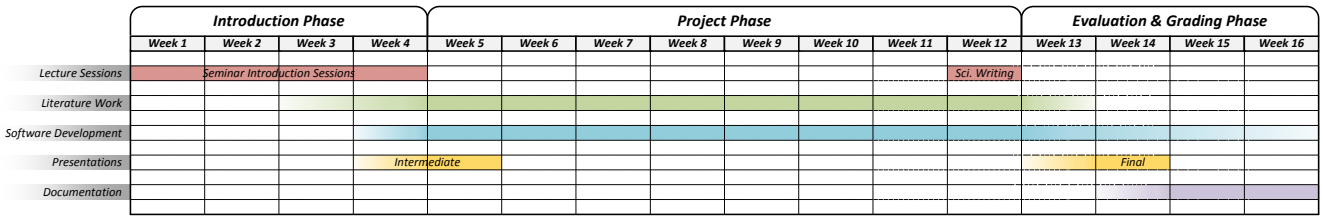

Figure 2: *The course is structured in mainly three phases: In the introduction phase multiple lectures are given to provide a common knowledge base. The project phase starts with literature work and an intermediate presentation of the students. This phase is dominated by software development and ends with the final presentation. In the final evaluation phase the students document their work.*

a survey. Administrative aspects, such as the signing of a non-disclosure agreement (NDA), are regulated, if necessary.

Introduction to Image Processing: In usually two to three sessions, the general concepts and techniques of image processing are covered on an introductory level. This includes image and video data representations, as well as point-based, neighborhood-based, and global processing techniques.

Basics of OpenGL ES and GLSL ES: These sessions cover specifics of the OpenGL ES and WebGL APIs as well as differences to desktop OpenGL versions, students may be familiar with. Exemplary implementations of image processing techniques that are available in the framework are used to illustrate the APIs.

Software Development Tools: These sessions introduce the Git code versioning system including issue writing, the branching concept, and merges in particular. In addition, the work flow of respective web platforms (e.g., Bitbucket or GitHub) are discussed. Further, IDEs such as Android Studio, are briefly presented, especially the build process and debugging functionality.

Software Framework: Finally, to steepen the technical learning curve for integrating OpenGL ES API and OpenGL ES shader programs with Android, a brief introduction to the rendering framework is given. It covers the framework's architecture, assets, and file formats.

For all students, the attendance of all lectures is mandatory. In order to respond to the students' potential foreknowledge gained in other courses, they are asked during the kick-off for possible topics they like to focus on.

2.4. Topic Assignment

Subsequent to the topics briefly pitched during first session, each student chooses three topics and submit them to the course instructor. In addition thereto, the students should submit device specifics (brand, version of operating system, OpenGL API version, etc.) to check whether the mobile devices are capable for implementation of the chosen topics. If this is not the case, the work group can lend a device to the respective student. The topic assignments are then performed by the instructors using the first-come-first-served principle. In exceptional cases, students are required to sign an NDA to protect intellectual property of partner institutions or companies. The results of the topic assignment is then presented to the class during the next session. Following to that, each student meets with the personal instructor.

In this one-on-one meeting between instructor and student, *must-have*, *should-have*, and *nice-to-have* features are defined for the topic resulting in a requirements document (similar to contracts or user stories) that serves as one basis for grading. These documents can also be prepared in advance. We observed that such kind of documents support especially undergraduate students in structuring their topic into milestones and features (cf. Section 2.6). Further, if required by the student, a recurring appointment for further one-on-one meetings during the course of the semester is defined.

The complexity of the project topics differs between bachelor and master courses. While topics for bachelor student mostly focus on using the framework by implementing shader programs and rendering pipelines (specified using XML) for image-processing effects, master students' topics partially include research questions [AAF16] (Section 3.1) that require to extend the framework's code base. In general, students are encouraged to define their own topics in collaboration with their supervisor [Rom13].

2.5. Intermediate Student Presentations

To enforce to start work early during the semester, students must prepare and deliver a brief intermediate presentation in front of the class and instructors. The presentation duration of ten minutes plus five minutes feedback should include: (1) motivation of the topic, (2) a structured overview of related work, and (3) a structured overview of their approach. To facilitate students to focus on the presentation contents and not its formatting or layout, LaTeX, Keynote, and PowerPoint presentation style templates are provided. All slides are distributed via LMS. Students are also encouraged to use the whiteboard for presentation. In addition to a projector, a digital clock showing the remaining time is displayed. The presentation material of all students is collected and made accessible to all students of this seminar subsequently to the presentation.

During feedback round, fellow students and instructors can give direct feedback on the chosen approach, used technology, and presentation skills. Based on experience, feedback on presentation skills are made at classmate level. Therefore, two students that are randomly selected must comment on the presenters performance. To facilitate grading of the intermediate presentation, feedback sheets (Figure 4), which are known to the students, are filed by instructors directly after the session.

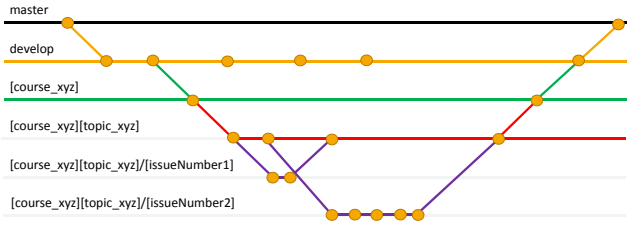

Figure 3: *Example of the course software branching concept. Each course instance has its own branch (green). Based on that, each student works in his / her own topic branch (red), which represents the parent of the specific branches (violet) for each issue to implement.*

2.6. Project Phase

This phase is the most labor-intensive time for the students and comprises basically three concurrent tasks: (1) *literature work*, (2) *software development*, and (3) attending *status meetings* on a regular basis, as well as broadcast communication.

Literature Work. An important didactic goal in higher education is to gain experience in proper literature work. Therefore, the students must create and maintain a collection of related and previous work. The results of this process are: (1) a bibliography file (e.g., for BibTeX) including searchable abstracts and linked source files (e.g., Digital Object Identifier (DOI) or local PDF files) for each resource, as well as (2) a brief bullet-point list referencing these, which are used in master courses for paper preparation (Section 2.9).

Software Development. The organization of the software development process is designed to be close to agile industrial software development [Wil12] with its respective tools. Each student works on their own *topic branch* [BA02] in the version control system (Figure 3). During the development process, for each identified feature an *issue* and a respective *feature branch* is created. Issues are described in the form of *user stories* [Coh04].

After completing a feature, the respective branch is merged into the topic branch by pull requests (PRs), which are reviewed by fellow students and (in the beginning) by the respective course instructors. This increases code quality, enables mutual learning, and keep the instructors "in the loop" [MKAH14, BCB*17]. The reviewing students are selected during introduction phase based on the thematic proximity and common factors with respect to the implementation. Code reviews follow a contemporary process mainly used in industry [RB13]. Participation in code reviews is mandatory. Finally, based on the specifications described in the user story of the respective issue and a *definition-of-done* [Wil12], the PR is than approved or declined.

To steepen the implementation learning curve, descriptive effect templates, shader program templates, and class templates are provided that demonstrate the respective naming schemes (coding style guides) and documentation principles (e.g., JavaDoc). Students are allowed and encouraged to reuse existing source code. Further, as commonly used in industry, a *continuous integration platform* executes build step after each push to their branch [Wil12]. To enforce good code quality a linter automatically checks code and comment style conventions in continuous integration.

Individual Status Meetings. To keep track of development and establish a forum for advise using direct communication between student and instructor, a (bi)weekly meeting shows to have been most effective. Depending of the progress and topics to discuss, the meeting duration is usually between 15 and 60 minutes. At the end of the meetings, a brief informal protocol is shared, used to pick-up communication for the subsequent meeting.

Information Broadcast & Discussions. During the course working phase, each student usually works individually. To support and facilitate communication between fellow students and to broadcast specific or important course information, two newsgroups are established in the beginning of each course: one open to instructors (*news channel*) and one for students only (*discussion board*). Changes in course dates etc. are communicated using a *course calendar*. Each enrolled student receives updates via email.

2.7. Final Student Presentations

Based on the experience made during the intermediate presentations (Section 2.5), the students present their results by giving a final talk of their achieved course results. The talk duration can vary: for example 20 minutes talk plus 10 minutes open discussion for bachelor students or 25 minutes plus 5 minutes for master students. For demonstration purposes, a competitive and common set of test images is used, in our case focusing on image stylization [MR16]. The remaining section covers presentation setup, grading, and feedback channels used.

Presentation Setup. In contrast to intermediate presentations, the presentation setup is more elaborated. For final presentations, the students must prepare (1) a *slide deck*, (2) an executable and deployable *prototype*, as well as (3) a demonstration *video*:

Slide Deck: The presentation should be prepared in terms of a scientific talk. Basically, the talk should be structured into distinguished sections: *introduction* (motivation, teaser, problem statement, and conceptual as well as technical challenges), *related work* (overview and classification), *concept*, *implementation* (pipeline integration, OpenGL specifics, architecture using unified modeling language (UML)), and *discussion* (performance evaluation, conceptual and technical limitations), *conclusions and future work*. Beside electronic presentation, the students are encourage to also use the whiteboard for communication. The slides are collected prior to presentations and distributed afterwards.

Prototype Application: We observed that the most effective way to present achieved results is by demonstrating features implemented in a prototypical mobile app. Therefore, students can prepare a deployable version of their app, which is then installed on a presentation device prior to the sessions. Thereby, we used a Google Chromecast for the demonstration setup.

Demonstration Video: In addition, students should prepare a short video (approximately 5 minutes in length) to demonstrate

Presentation Feedback Form

(Based on the "HPCC Seminar Presentation Feedback Form", Department of Computer Science, University of Minnesota)

Speaker:	*Date:*
Talk title:	
Evaluator:	*Familiarity with the topic: (++ , + , O , - , -- , ?)*

1. Overall Structure	
a) Was the presentation well organized?	(++ , + , O , - , -- , ?)
b) Did each slide make the intended point?	(++ , + , O , - , -- , ?)
++ for slide: -- for slide:	
c) Were the slides well presented? (E.g. color, fonts, space, etc.)	(++ , + , O , - , -- , ?)
++ for slide: -- for slide:	
2. The Introduction	
a) Did the talk have a distinct introductory section?	(++ , + , O , - , -- , ?)
b) Did the introduction clearly state and explain the problem?	(++ , + , O , - , -- , ?)
c) Did the introduction attract the audience curiosity?	(++ , + , O , - , -- , ?)
3. The Body	
a) Was the presenter's center approach clearly stated?	(++ , + , O , - , -- , ?)
b) Was the proper amount of detail presented?	(++ , + , O , - , -- , ?)
c) Were the results adequately justified?	(++ , + , O , - , -- , ?)
4. Conclusion	
a) Did the talk have a distinct conclusion?	(++ , + , O , - , -- , ?)
b) Did the conclusion summarize the key aspects of the talk?	(++ , + , O , - , -- , ?)
c) Were the take home points clearly stated?	(++ , + , O , - , -- , ?)
5. Questions / Answers	
a) Were the questions (if occurred) handled properly?	(++ , + , O , - , -- , ?)
6. Presentation Style	
a) Was the presentation easily audible and visible?	(++ , + , O , - , -- , ?)
b) Did the speaker make eye contact?	(++ , + , O , - , -- , ?)
c) Did the speaker respond to the audience's needs?	(++ , + , O , - , -- , ?)
d) Were all terms properly defined before they were used?	(++ , + , O , - , -- , ?)
7. Free Space for additional Comments	

(Please use backpage.)

Figure 4: *Presentation feedback form used for structured feedback on intermediate and final presentations.*

features to be used as a fall-back solution and alternative to the prototype. Videos from mobile devices can be easily captured from the IDE (Android Studio).

To support students to keep track of their timing during performance, and to simulate a conference-alike situation, an additional timer can be displayed. As an alternative, the session chair raises leaflets with timings (15, 10, 5, and 1 minute). Further, to raise additional motivation, a *best project award* is sponsored by the working group. The three possible winners (obtaining a voucher of different price categories each) are chosen by the number of votes. The complete auditorium have three votes for best presentation. The complete work group (including PhD students and senior research staff) attend these presentations.

Presentation Grading. Since the final presentation significantly contributes to the final mark, it is important to make the grading process as transparent as possible to the students. Therefore, each instructor fills out a *presentation grading form* (Figure 4) during the respective presentation. Subsequent to all presentations, the advisors gather the forms and perform grading. The results of this process is documented accordingly for students interested in their performance.

Presentation and Student Feedback. Proper presentation feedback is important to cultivate and improve communication and feedback culture among students and advisors. Therefore, at the end of each presentation, a brief and open *feedback round* is conducted. In general, these feedback rounds are moderated by one of the advisors. Two students are selected randomly to give feedback. After the presentation sessions, the instructor gives its respective students elaborated feedback in an one-on-one meeting.

During the last course session, students are encouraged to give direct voluntary feedback to the course instructors to highlight aspects for improvement or performance. These "lessons learned" are documented and shared among the working group to facilitate subsequent efforts of instructors to improve the teaching concepts and to adapt to different student generations. This is performed in addition to using an institution-wide evaluation platform that enables anonymous feedback and instructor assessment. Section 3 discusses these evaluations and its results in more detail.

2.8. Final Code Submission

The final code submission is performed by creating a *final pull request* (FPR) of the complete student's topic branch into the course branch. This includes review of submitted code and commenting where applicable. The student has one week time to correct or dispute comments. The instructors check-out the individual branches, compiles, deploys and test the code. Based on the results, the respective instructor *approves* or *decline* the FPR. Following to that, the grading is based on different criteria, such as (1) effectiveness of the approach, (2) achieved performance, and (3) adherence to code style guidelines.

In addition thereto, students deliver a brief overview page of the implemented technique to facilitate conceptual understanding and code reuse of their achievements. It includes: (1) a rendering pipeline overview, (2) notations, ranges, and default values of parameters, as well as (3) a description of compatibility issues. This document is stored in the respective versioning system as Markdown! document (RFC 7763).

2.9. Paper Submission

In master courses, students have to submit a research paper describing their results in a scientific manor within up to 4 to 5 pages (short paper), preferably using the English language. A didactic goal is to introduce students to scientific writing and to gain experiences with tools such as LaTeX, BibTeX, including bibliography data basis such as JabRef, Mendeley or similar. The paper structure and presentation should adhere to scientific writing standards.

To ease the writing process, a common conference style template is handed out to the students (e.g., the LaTeXAuthor Guidelines for Eurographics proceedings). If requested by the students, an additional lecture slot on "Scientific Writing" is given by the course instructors, which is often the case for first year master students. The paper submission process for grading comprises three steps: (1) *paper preparation*, (2) *paper review*, and (3) *paper grading*.

Paper Preparation. Paper preparation starts subsequently to the final presentation sessions. Already at the beginning of the master course, related work on scientific writing are provided to the students.For paper preparation, the students start by drafting a preliminary paper structure, writing an abstract, and by creating stubs

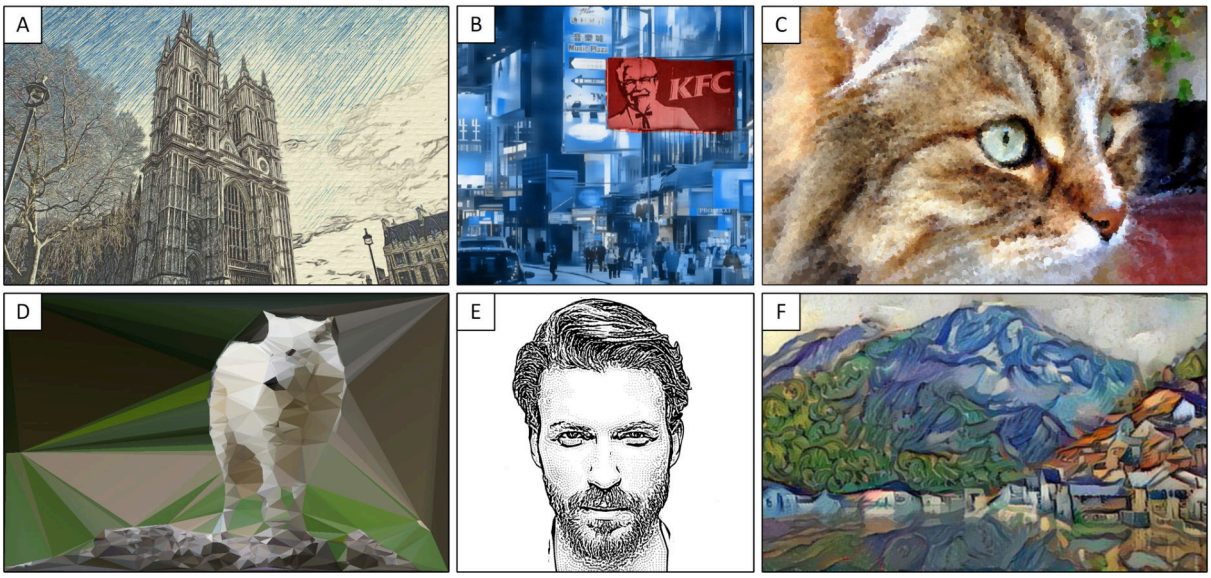

Figure 5: *Examples of successful student projects (top row: bachelor students' results, bottom row: master students' results).*

for images and figures. At this point, students are strongly encouraged to integrate the (content) feedback from final presentation into their papers. The submission deadline is set by LMS. Papers that are submitted too late are not considered for grading or the subsequent *review cycle*.

Paper Review & Grading. Subsequent to paper preparation, the students must submit their paper draft version into a *review cycle*. Two students that are assigned by the course instructor and that are not the same as the respective PR reviewer (cf. Section 2.6) review the paper of a fellow student within one week. In order to organize and structure the review process, a review form similar to these of computer graphics conferences such as Eurographics or SIGGRAPH are used. The review results are send back to the student as annotated PDF file. The final paper grading for each student is performed by the respective course instructor using basically the same process as the paper review stage.

3. Evaluation and Discussion

We evaluated the presented teaching concept during four project seminar implementations—i.e., two bachelor and two master seminars respectively (one master seminar is ongoing at the time of writing)—by means of working examples and feedback results.

3.1. Student Working Examples

Figure 5 shows some examples of recent student projects. The top row are results achieved by bachelor students: (A) a pencil hatching app based on orientated tonal art maps [WPFH02], (B) an effect mimicking the style of Sheng-Qi based on bilateral filtering [TM98], (C) interactive, adjustable painterly rendering using an example-based rendering technique similar to [Her98]. The bottom rows depict working examples achieved by master students: (D) a low-poly effect, (E) a real-time implementation of feature-guided image stippling [KSL*08], and (F) semantic-aware style transfer based on generative convolutional neural networks [ZD17].

Out of 37 student projects, four contributed to research papers for international conferences. Two projects have led to winning two "Best Demo Awards" and a *Best Paper Award* at renowned international symposia. Furthermore, one student project received a student speaker invitation to the *Ada Lovelance Festival* and one was submitted to a national media award. Eight students continued their project as student assistants to further contribute to the development of the framework, and more than ten students participated in follow-up projects or lectures.

3.2. Evaluation Results

The student grading results indicate a good learning success for the finished bachelor ($\mathcal{A}$ and $\mathcal{B}$) and master ($\mathcal{C}$) courses. Students of course $\mathcal{A}$ were graded overall with 1.2 (out of the range 1–5, i.e., 1 means very good and 5 means the seminar has not been passed) with a standard deviation of $\sigma = 0.23$. Students of course $\mathcal{B}$ were graded overall with 1.7 with a standard deviation of $\sigma = 0.67$. The master's course students $\mathcal{C}$ were graded 1.6 with a standard deviation of $\sigma = 0.73$.

The following anonymous, quantitative and qualitative student feedback has been obtained using a course evaluation platform. Students are able to give their feedback prior to the grading process, and the results are published after the course has been finished. Qualitative feedback is available to people that have been evaluated and to the person responsible for the course.

Quantitative Student Feedback. Course $\mathcal{A}$ was graded overall with 1.4 (out of the range 1–5, with 1 means very good) by 71 % of the bachelor students (5 out of 7 participants) with a standard

Table 1: *Overview of quantitative course evaluation results. The grades (*1–5*) are categorized into* very good *(green:* 1.0–1.6*),* good *(orange:* 1.7–2.3*), and* satisfying *(red:* 2.4–3.3*).*

Aspect	Bachelor Course ($\mathcal{A}$)	Bachelor Course ($\mathcal{B}$)	Master Course ($\mathcal{C}$)
Participants:	7	16	14
Course content:			
I learned a lot in the course.	2.0	1.6	1.6
The content of the course was well structured.	1.2	1.6	1.9
The course deepened my knowledge in this topic.	1.2	1.3	1.6
Learning Material:			
The provided learning material was sufficient.	1.0	1.8	1.6
The provided learning material was useful.	1.2	1.7	1.5
Project:			
I enjoyed the project.	1.2	1.7	1.7
The expenditure of time was appropriate.	1.6	3.0	3.0
I learned something from the project.	1.6	1.6	1.6

deviation of 0.4. Course $\mathcal{B}$ was graded overall with 1.7 by 62.5 % of the bachelor students (10 out of 16 participants) with a standard deviation of $\sigma = 0.6$. The master's course $\mathcal{C}$ was rated 1.6 by 50 % (7 out of 14 participants) with a standard deviation of 0.5. Table 1 shows grades by participants with respect to specific evaluation questions. In general, the course was positively rated, with an exception regarding the time that has to be invested. Compared to course $\mathcal{A}$, course $\mathcal{B}$ and $\mathcal{C}$ required the students to invest more time, since writing an additional scientific paper was required or a more complex task had to be solved. Students state that this yields problems in time management, which subsequently impacts other factors such as motivation and joy. However, the same is observed in lectures and seminars of other work groups as well.

Qualitative Student Feedback. Besides the quantitative results, students are able to give qualitative feedback in form of responses to general questions, which is presented in the following.

The question *"What were the strengths of the course?"* was addressed as follows. Some students rate the available programming framework as practical, since it steepens the learning curve and enables fast and early prototypes, and thus, more feedback iterations. In general, the use and application of Git and Bitbucket as well as the accompanying software development process were received positive. Further, the possibility to lend a mobile testing device was received overall positive. Although, paper writing was not part of the concept for bachelor courses, some students were interested to do so. Most students were satisfied with their topics and make use of possible synergy effects between implementations.

The question *"How could the course be further improved?"* received the most student feedback. In general, master students find the efforts of a software project, two presentations, one paper, and the additional pull requests (PRs) as too much for 6 ETCS points. Especially PRs were identified by the students as a possible critical aspect, since some PR were not processed in time by fellow students. A mentioned cause for that is the missing opportunity for team work, which limits the motivation to review (considered as an "additional", non-integral part of the work). From a technical point of view, relatively slow deployment times (up to 1–2 minutes) to the mobile device were generally considered negative, since it prevents quick "trial and error" iterations in programming. This especially arises due to the lack of OpenGL ES capabilities in the Android Studio emulators (limited to OpenGL ES 2.0). A minority of students also missed a theoretic introduction to concepts regarding their topics and more in-depth discussion of image operations during the introduction lectures.

3.3. Discussion & Lessons Learned

Overall, we received positive feedback for this course format over the last five semesters and registered an increase of students. Only one (master) student out of a total of 37 did not successfully finish the course. In the last seminar, 16 bachelor students were enrolled, which represents a limit to the instructor work loads. Thus, a successful accomplishment of such project seminar usually requires more than one course instructor. Even for up to three instructors, it results in a relatively high (but usually predictable) workload. We strive towards continuously offering this project seminar in future terms by alternating between bachelor and master courses. However, this will require changes in the concept especially for the master courses. We identified several aspects to support the student's time management for the seminar: Concerning the schedule, the intermediate presentation should start early during introduction phase and regular group meetings for direct communication should be introduced. Instead of reducing the student's work load directly, e.g., by trading final presentation against paper writing, we chose to allow team work with a maximum of two students per team, which is evaluated in the current semester.

Despite the focus of this paper, the presented concept is applied during recent years in several bachelor and master courses. Although the described concept seems cumbersome, it showed to have some major advantages for the student's education and with respect to code reuse. Using a common software framework and working with PRs, a higher code quality could be obtained, which reduces possibly future refactoring time. The increased effort for students and instructors for this course format yields the question of the direct *benefits*. These can be summarized with respect to the students, instructors, and collaborating companies as follows:

Students: The benefits for students are diverse. First, we observed that the gained experiences in presentation and paper writing skills have a positive effect on the resulting quality of their bachelor and master thesis. Further, early introduction to scientific working principles raises the motivation to participate in subsequent teaching and research activities, for example by contributing to research papers or working as student assistants. The usage of industry-proven software development processes and the complex common software framework enables students to work as professional software developers or interns for the collaborating companies. Furthermore, students can use the prototypes to demonstrate their results to friends and family outside the academic environment, which increases motivation for this topic.

Teachers: Teaching personal can benefit from various seminar results. The early prototypical implementations of image processing techniques and algorithms often represent first feasibility studies of research ideas. These can be used to support future research and publications as well as to facilitate funding activities. Further, an early effort in teaching expenses is counterbalanced

by reduced teaching overhead during the subsequent supervision of bachelor, master, and PhD theses. Furthermore, the resulting code artifacts and documents can be easily integrated into the existing framework and for future research and teaching.

Collaborating Companies: Besides proof-of-concept implementation that can be used to communicate new ideas, inspired and specifically educated students are a major benefit for collaborating companies. These can represent future student trainees as well as employees, which are familiar with challenges and fields of engineering that are of particular interest to a company.

4. Conclusions and Future Work

This paper describes a concept and implementation details for a project seminar course in the computer graphics domain applied to teach the implementation of real-time image processing techniques and algorithms for mobile devices. The seminar focuses on combining several teaching goals: gain hands-on experiences in OpenGL ES or WebGL in combination with industry-wide software processes and research skills such as giving presentations and paper writing. The presented concept and the evaluation of its implementation partially answers questions to the much discussed topic of undergraduate research and how it can be integrated into undergraduate degree programs [AAF16]. For future work, a major question is how to break down the presented concept for a massive online-enrolled courses (MOOC) on image-processing programming for mobiles devices, or computer graphics courses in general, which represents an increasing demand today [KH16, Bou16].

Acknowledgments

We thank all students and researches that participated in the seminar for their work and extensive feedback. This work was funded by the Federal Ministry of Education and Research (BMBF), Germany (AVA, 01IS15041).

References

[AAF16] ANDERSON E. F., ADZHIEV V., FRYAZINOV O.: Aiming High: Undergraduate Research Projects in Computer Graphics and Animation. In *Proc. Eurographics - Education Papers* (2016), Santos B. S., Dischler J.-M., (Eds.), The Eurographics Association. 1, 3, 8

[BA02] BERCZUK S. P., APPLETON B.: *Software Configuration Management Patterns: Effective Teamwork, Practical Integration*. Addison-Wesley Longman Publishing Co., Inc., 2002. 4

[BCB*17] BOSU A., CARVER J. C., BIRD C., ORBECK J., CHOCKLEY C.: Process Aspects and Social Dynamics of Contemporary Code Review: Insights from Open Source Development and Industrial Practice at Microsoft. *IEEE Transactions on Software Engineering 43*, 1 (2017), 56–75. 4

[Bou16] BOURDIN J.-J.: MOOCs in Computer Graphics. In *Proc. Eurographics - Education Papers* (2016), Santos B. S., Dischler J.-M., (Eds.), The Eurographics Association. 8

[CC05] CLAYPOOL K., CLAYPOOL M.: Teaching software engineering through game design. *SIGCSE Bull. 37*, 3 (June 2005), 123–127. 2

[Coh04] COHN M.: *User Stories Applied: For Agile Software Development*. Addison Wesley Longman Publishing Co., Inc., 2004. 4

[CPAM08] CAPIN T., PULLI K., AKENINE-MÖLLER T.: The State of the Art in Mobile Graphics Research. *IEEE Computer Graphics and Applications 28*, 4 (July 2008), 74–84. 1

[Dev13] DEV K.: Mobile Expressive Renderings: The State of the Art. *IEEE Computer Graphics and Applications 33*, 3 (May 2013), 22–31. 1

[DFH02] DEMUTH B., FISCHER M., HUSSMANN H.: Experience in early and late software engineering project courses. In *Proc. Conference on Software Engineering Education and Training* (2002), CSEET '02, pp. 241–248. 2

[DT03] DOUGIAMAS M., TAYLOR P. C.: Moodle: Using learning communities to create an open source course management system. In *Proc. of the EDMEDIA 2003 Conference* (2003). 2

[DTD17] DÜRSCHMID T., TRAPP M., DÖLLNER J.: Towards Architectural Styles for Android App Software Product Lines. In *Proc. MOBILESoft* (2017), pp. 58–62. 2

[FWW13] FINK H., WEBER T., WIMMER M.: Teaching a modern graphics pipeline using a shader-based software renderer. *Computers & Graphics 37*, 1–2 (Feb. 2013), 12–20. 1

[GW06] GONZALEZ R. C., WOODS R. E.: *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., 2006. 2

[Her98] HERTZMANN A.: Painterly Rendering with Curved Brush Strokes of Multiple Sizes. In *Proc. SIGGRAPH* (New York, NY, USA, 1998), SIGGRAPH '98, ACM, pp. 453–460. 6

[KH16] KAPLAN A. M., HAENLEIN M.: Higher education and the digital revolution: About MOOCs, SPOCs, social media, and the Cookie Monster. *Business Horizons 59*, 4 (2016), 441–450. 8

[KSL*08] KIM D., SON M., LEE Y., KANG H., LEE S.: Feature-guided Image Stippling. *Computer Graphics Forum 27*, 4 (2008), 1209–1216. 6

[MKAH14] MCINTOSH S., KAMEI Y., ADAMS B., HASSAN A. E.: The Impact of Code Review Coverage and Code Review Participation on Software Quality: A Case Study of the Qt, VTK, and ITK Projects. In *Proc. Working Conference on Mining Software Repositories* (2014), MSR '14, pp. 192–201. 4

[MR16] MOULD D., ROSIN P. L.: A Benchmark Image Set for Evaluating Stylization. In *Non-Photorealistic Animation and Rendering* (2016), The Eurographics Association. 4

[RB13] RIGBY P. C., BIRD C.: Convergent Contemporary Software Peer Review Practices. In *Proc. Foundations of Software Engineering* (2013), pp. 202–212. 4

[Rom13] ROMERO M.: Project-Based Learning of Advanced Computer Graphics and Interaction. In *Proc. Eurographics - Education Papers* (2013), Bourdin J.-J., Cerezo E., Cunningham S., (Eds.), The Eurographics Association. 3

[SDT*16] SEMMO A., DÜRSCHMID T., TRAPP M., KLINGBEIL M., DÖLLNER J., PASEWALDT S.: Interactive Image Filtering with Multiple Levels-of-control on Mobile Devices. In *Proc. SIGGRAPH ASIA Mobile Graphics and Interactive Applications* (2016), SA '16, pp. 2:1–2:8. 2

[She11] SHESH A.: High-Level Application Development for non-Computer Science majors using Image Processing. In *Proc. Eurographics - Education Papers* (2011), Maddock S., Jorge J., (Eds.), The Eurographics Association. 2

[TM98] TOMASI C., MANDUCHI R.: Bilateral Filtering for Gray and Color Images. In *Proc. International Conference on Computer Vision* (1998), ICCV '98, pp. 839–846. 6

[TMB14] THABET R., MAHMOUDI R., BEDOUI M. H.: Image Processing on Mobile Devices: An Overview. In *Proc. International Image Processing, Applications and Systems Conference* (2014), pp. 1–8. 1

[Wer13] WERNER M.: Teaching Graphics Programming on Mobile Devices. *J. Comput. Sci. Coll. 28*, 6 (June 2013), 125–131. 2

[Wil12] WILLIAMS L.: What Agile Teams Think of Agile Principles. *Commun. ACM 55*, 4 (Apr. 2012), 71–76. 4

[WPFH02] WEBB M., PRAUN E., FINKELSTEIN A., HOPPE H.: Fine Tone Control in Hardware Hatching. In *Proc. Symposium on Non Photorealistic Rendering* (June 2002), pp. 53–58. 6

[ZD17] ZHANG H., DANA K. J.: *Multi-style Generative Network for Real-time Transfer*. Tech. Rep. 1703.06953, arXiv, 2017. 6